

Arduino-based SWR analyser

In this project, a simple hardware SWR bridge is used as a measurement front end for the open-source Arduino development platform.

The user can develop their own solution based on the measurement software developed for the project, which is in the public domain and available on Github. It is not a complete, finished and ready-to-roll project: my hope is that this will stimulate fellow amateurs to build their own boards and experiment with the basic software to enhance the features that the analyser can provide.

In its basic form the analyser gives an accurate measurement of the SWR (in a 50Ω system) and impedance of an aerial at any frequency to >70MHz. Coaxial cable length can be measured and the resonant frequency of an aerial can be tracked. The analyser has the capability to be controlled from a program on a PC to allow data logging and/or graphical analysis of results.

SWR is a measure of the mismatch between the load and the design impedance of the system and gives an indication of the ratio of the impedances. This is a scalar measurement and does not give any information about the capacitive or inductive reactance except for its impedance at the test frequency. Judgement of whether a reactive load is capacitive or inductive requires interpretation of the measurement.

Project history

A desire for an inexpensive SWR analyser and an interest in direct digital synthesis (DDS) modules led to the start of this project. The AD9851 DDS is capable of synthesising a sine wave RF signal up to 50MHz or more with superb accuracy and stability. The cheaper AD9850 is good to 30MHz, but a small software change is needed to accommodate a slight difference in control word for the latter device. Both are available as assembled modules on eBay for only a few pounds.

The Arduino open-source platform uses a simplified C language environment for program development [1]. The ready availability of low cost digital hardware and a free software development platform to complement it allowed rapid evaluation of a proof-of-concept design.

The Arduino Uno board is based on an ATmega328, which has 20 inputs/

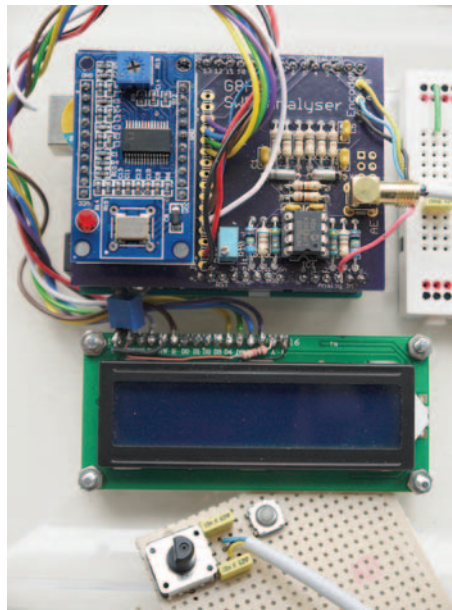


PHOTO 1: The prototype board with piggybacked DDS module (left side of dark blue PCB).

outputs. Up to 5 inputs can be configured as analogue, feeding an internal analogue to digital converter (ADC) and two can become external interrupt inputs, which is perfect for this sort of project. The Uno motherboard has socket headers with all the signals brought out to allow a development board to be piggy-backed. This gives a neat solution with plenty of resources to control the DDS, an LCD display and a rotary shaft encoder with push switch to select menu choices.

What's needed to make an SWR analyser?

For the majority of day-to-day practical uses, the amateur only needs to measure the SWR of the aerial at the chosen frequency, or to determine the frequency of lowest VSWR when tuning an aerial. In the past, this information was provided by a VFO, bridge, diode detector and moving coil meter: now, for a similar cost we can have a tool that can be programmed accurately to give greater flexibility and functionality (if the user so desires).

In the Wheatstone Bridge (see **Figure 1**), V_r gives an indication of mismatch and would traditionally have driven a moving coil meter. The ready availability of low-cost microcontrollers such as the Arduino allows us to convert V_s and V_r to digital values and process them. Most commercial aerial analysers for the amateur market use some form of the Wheatstone Bridge.

A calculation involving V_s and V_r will reveal the SWR:

$$\text{VSWR} = (|V_s| + |V_r|) / (|V_s| - |V_r|)$$

where $|V|$ is the absolute value of the voltage, ie the magnitude.

If the bridge is in balance ($Z = 50\Omega$),

$V_r = 0$ and $\text{VSWR} = 1$

If the load is open circuit, $|V_r| = |V_s|$ and $\text{VSWR} = (2 \times |V_s|) / 0 = \infty$

If the load is short circuit, $|V_r| = |V_s|$ and $\text{VSWR} = (2 \times |V_s|) / 0 = \infty$

A diode RF envelope detector (as shown in **Figure 1**) allows the magnitude of the RF signal to be measured accurately down to millivolt levels (when lightly loaded, eg buffered by a high impedance amplifier circuit). At low

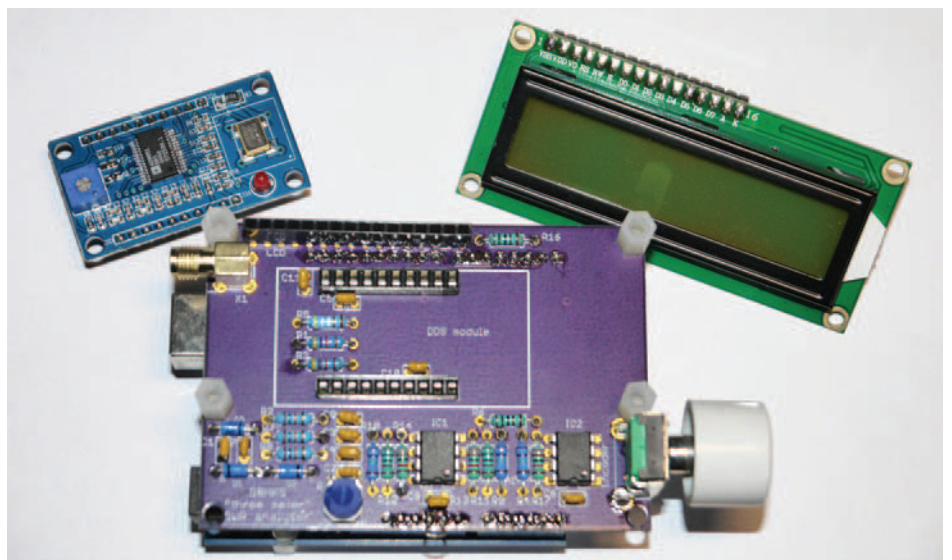


PHOTO 2: Second prototype. Main PCB (centre), DDS (top left) and display.

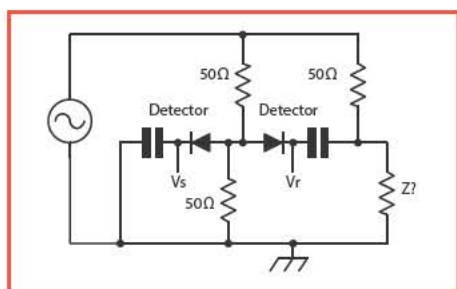


FIGURE 1: RF Wheatstone Bridge.

signal levels the diode exhibits a logarithmic response, but this can be compensated for in software. A comparative measurement of the I/V curves of 1N914 silicon small signal, 1N34 gold bonded germanium and BAT43 Schottky diodes revealed that the modern Schottky diode performs very well, with a forward voltage drop of only 23mV at $<1\mu\text{A}$. And, unlike the older 1N34, BAT43 diodes are readily available!

The detector that measures V_r measures the magnitude of the voltage difference between the two arms of the bridge, and truly floats at AC whilst producing a ground referenced DC signal through the return path consisting of the bridge resistors.

Each low-level detector output is buffered by a high input impedance operational amplifier circuit with sufficient gain (around 20) to give a useful output in the 5V range of the ADC on the AVR processor chip. Following digitisation, the non-linearity of the diode detector is compensated for in the software to improve the accuracy of the results.

The same VSWR will result from both high and low load impedances: SWR 2:1 for both 100Ω and 25Ω, SWR 3:1 for 150Ω and 17Ω, etc. The addition of a third detector to measure the voltage across Z gives us the opportunity to determine whether the load impedance is greater or less than 50Ω. If V_z is low, the impedance must be low; a high V_z is the result of a high impedance.

From breadboard to prototype

Work on a basic two detector bridge design using a breadboard proved that the concept

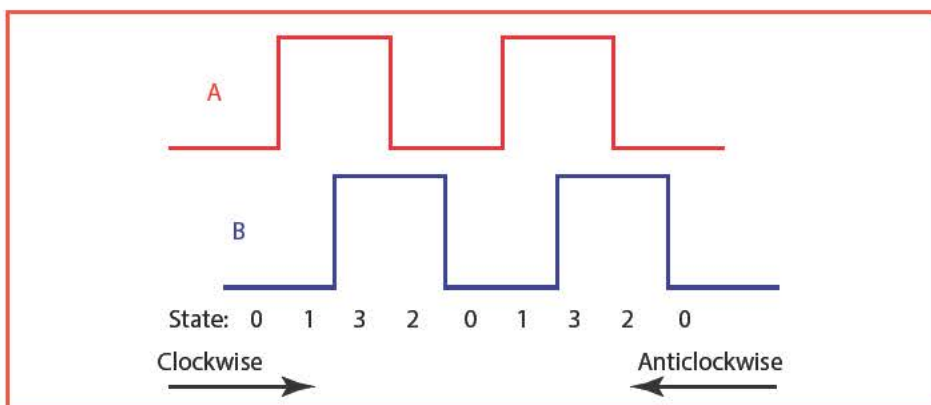


FIGURE 2: Level transitions of quadrature encoder.

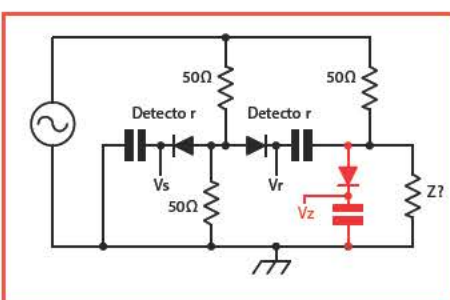


FIGURE 3: Addition of detector for V_z (in red).

was sound and allowed me to write the base software to control the DDS, accept input from the shaft encoder and drive a two line LCD display for menu selection and to display results. I then designed a prototype PCB to allow further development in a more controlled environment.

OSH Park [2] in the USA offer an excellent service for cost-conscious experimenters and provided me with 3 PCBs for \$5 per square inch, delivered in roughly 2-3 weeks. The painless process involves submitting the design file from a PCB CAD package to their website and paying by PayPal. The designer is kept fully informed on the progress of the PCB, and following the announcement that it has shipped the excitement grows! (Other prototype PCB manufacturers offer comparable terms – Ed).

The board is designed to plug directly into an Arduino Uno. The Arduino requires a stable

5V power supply, which can either come from a PC connected to the Arduino's USB port or from an external low voltage DC supply (eg a 12V wall wart or batteries) through the on-board regulator. The Arduino has a USB serial link to an attached PC that is used for downloading the program to the Arduino as well as being a communications link to allow data logging and debugging when the analyser is in action.

In Photo 1 the DDS module can be seen piggybacked on the left hand side of the prototype PCB, which itself is plugged into the (hidden) Arduino Uno development board.

The DDS is controlled through a Serial Peripheral Interface (SPI), which uses a clock, data in and data out signals to transfer the control word between the processor and the slave DDS device at very high speed. The control word of the DDS is programmed with a 32 bit number to define the required frequency, which can be set to an accuracy of 0.04Hz(!)

The quadrature shaft encoder uses two signals (A and B) that alternately go high/low/high as the shaft is rotated, but with a lag between them. Figure 2 shows how this works.

The order in which the A/B quadrature signals are received is interpreted by the processor to give the direction of rotation and keep count of the relative position of the shaft. Using interrupts to detect the level changes allows the processor to continue with foreground tasks whilst the encoder is being rotated: each increment of the encoder creates an interrupt that calls the service subroutine, only taking a few microseconds to update the shaft position. My shaft encoder interrupt service routines were developed from code originally written by Beric Dunn, K6BEZ.

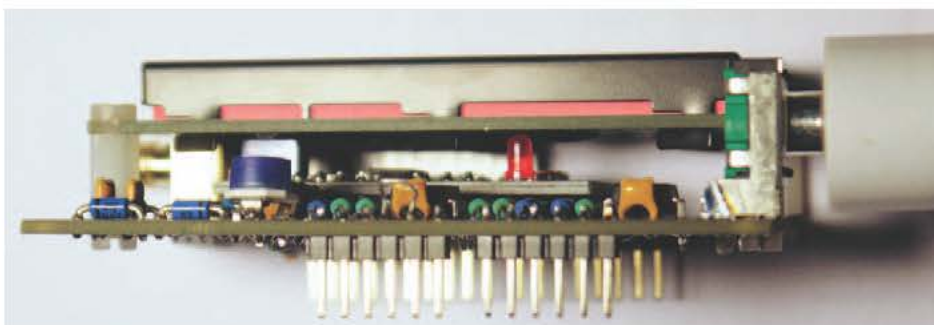


PHOTO 3: The stacked PCB, DDS and display.



PHOTO 4: Analyser with 50Ω termination.

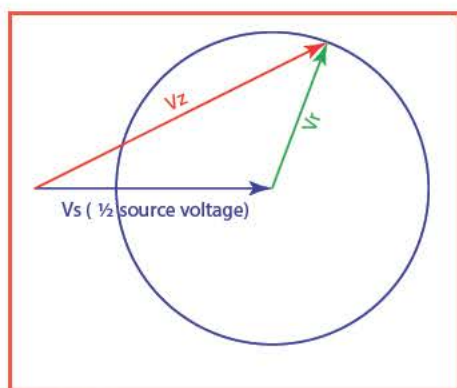


FIGURE 4: Vector triangle for a reactive load.

As the shaft is rotated, the processor counts with the transitions and increments or decrements the position count, which serves to index through menu options or control the DDS frequency. A separate switch contact made by pushing in the shaft of the encoder signals to select the current option. The menu structure is discussed later.

The success of the first board, plus the limitations of having the display and shaft encoder connected by cables, prompted the design of a second PCB (Photo 2, Photo 3 and Photo 4). Originally, SMA connectors were chosen because of their small footprint, but were replaced with BNC in the latest revision.

As mentioned earlier, the simple two-detector measurement of SWR has the limitation that it cannot distinguish between a low or high impedance load, both of which exhibit the same mismatch (eg 25Ω and 100Ω both give SWR of 2:1). In order to judge the magnitude of the load impedance, another detector is needed to measure the voltage V_z developed across the load, as seen in Figure 3, and this was added on the new board. A low value for V_z implies a low impedance and a high V_z a high impedance.

Note that all the voltage measurements are magnitudes and convey no information about phase shift in a reactive load, but the measurement of V_z gives us a clue to the magnitude of the reactance by solving the

triangle of voltage magnitudes formed from V_s , V_r and V_z (see Figure 4). The core code is shown in Figure 5.

The story so far

The hardware is at a stage now whereby measurements of V_s , V_r and V_z can be made and digitised, there is a control interface consisting of shaft encoder, switch and LCD display and a DDS delivers RF to the bridge at up to about 50MHz or more, with precise control.

The rest of the project is software, which is open-source. At present, the menu system (Figure 6) allows the following user control:

- Select any of the HF bands as a base to manually control the frequency of measurement and display VSWR and load impedance
- Measure the length of an open circuit coaxial cable
- Find and track the frequency of best match (useful for tuning an aerial).

```
// VSWR
void VSWR() {
  double Vs=0;
  double Vr=0;
  double Vz=0;
  double VSWR = 0; // VSWR = (1+Γ)/(1-Γ) = (Vf+Vi)/(Vf-Vi)

  // Read the forward and reverse voltages
  Vr = analogRead(REV_analogue_pin);
  Vs = analogRead(FWD_analogue_pin);
  Vz = analogRead(ZL_analogue_pin);
  if (Vr >= Vs) {
    Vr = Vs - 1;
  } // To avoid a divide by zero or negative VSWR

  // Calculate VSWR
  VSWR = (Vs+Vr)/(Vs-Vr);

  // Linearise VSWR
  VSWR = 1+9.55*log10(VSWR);

  // Calculate load impedance
  if (Vz < Vs) { // ZL is low
    ZL = 50/VSWR;
  } else { // ZL is high
    ZL = 50* VSWR;
  }
}
```

FIGURE 5: Core code.

Next steps

There are applications for the PC written in Python script (eg by PA20HH) that provide a graphical interface to display measurement results from the likes of this board, with control and data being exchanged over a serial USB connection. The next work to be done will involve additional software for the SWR analyser to accept connection to and control by the PC to provide a remote measurement head that can operate via a convenient USB connection.

As mentioned earlier, the impedance of a capacitive load decreases with frequency, so V_z for a capacitive load would also decrease (and the opposite is true for an inductive load). Therefore a trick to determine the sign of the reactive component of the load is to have the software solve the triangle to calculate the magnitude and phase of Z ? and then 'jiggle' the measurement frequency and determine the direction of change (and hence sign) of the complex part of V_z .

Including and beyond this, development of the software is open to the user. I will be very interested to hear what you do.

Websearch

- [1] www.arduino.cc
- [2] www.oshpark.com

Further reading

By coincidence, this month's Homebrew takes a separate look at the RF Wheatstone bridge see p36 Measuring RF Impedance Using the Three Meter Measuring Bridge, Peter Dodd, G3LDO, *ARRL Antenna Compendium*, 1995 Three Meter Method, Peter Dodd, G3LDO, *RadCom* September 2005 Amateur Measurement of $R+jX$, Doyle Stradlund, W8CGD, *QST* June 1965

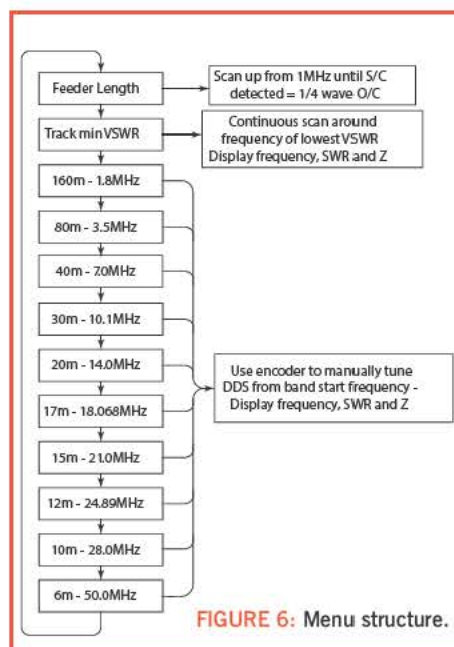


FIGURE 6: Menu structure.